A New SAT Encoding of the At-Most-One Constraint

Jingchao Chen

School of Informatics, Donghua University
2999 North Renmin Road, Songjiang District, Shanghai 201620, P. R. China
`chen-jc@dhu.edu.cn`

Abstract. In this paper, we study how to encode the *at-most-one* (AMO) constraint in conjunctive normal form (CNF). The AMO constraint means that at most one out of n propositional variables is allowed to be true. We present a new AMO encoding that improves on the existing one. The logarithmic bitwise AMO encoding by Frisch et al. requires $\log n$ auxiliary variables and $n \log n$ clauses. Sinz's sequential AMO encoding requires $n-1$ auxiliary variables and $3n-4$ clauses. Our recursive 2-product AMO encoding requires $2\sqrt{n}+O(\sqrt[4]{n})$ auxiliary variables and $2n+4\sqrt{n}+O(\sqrt[4]{n})$ clauses, which is fewer than the known best one. In terms of total number of literals appearing in the clauses, our encoding is the best, since it requires $4n+8\sqrt{n}+O(\sqrt[4]{n})$ literals, while Sinz's encoding and the logarithmic bitwise AMO encoding do $6n-8$ and $2n \log n$ literals, respectively.

Keywords: SAT encoding, At-Most-One constraint, encoding complexity.

1 Introduction

The *at-most-one* (AMO) constraint is a special case of cardinality constraints, which expresses that at most one out of n Boolean variables is allowed to be true. Because in recent years SAT solvers have made tremendous progress, many real-world problems are solved by encoding them in conjunctive normal form (CNF), i.e., converting to SAT problems, for example, EDA, artificial intelligence, crypt-analysis, planning, equivalence checking etc. Encoding a problem in CNF is called SAT Encoding also. SAT Encodings of many real-world problems require often an encoding of the AMO constraint. For example, the encodings of cardinality constraints [2], planning [2], partial Max-SAT [3] and Mixed Horn Formulas [4] make use of an AMO encoding. Therefore, devising a good AMO encoding is very important.

Let $X = \{x_1, x_2, \ldots, x_n\}$. The standard SAT encoding of the AMO constraint is the following.

$$\text{AMO}(X) \equiv \{\overline{x_i} \vee \overline{x_j} | x_i, x_j \in X, i < j\}$$

This AMO encoding requires $n(n-1)/2$ clauses. This is not an effective encoding, since it uses very many clauses. To reduce the number of clauses, based on sequential counters of the cardinality $x_1 + x_2 + \cdots + x_n \leq 1$, Sinz [1] introduced an AMO encoding as follows.

$$\text{AMO}(X) \equiv (\overline{x_1} \vee a_1) \wedge (\overline{x_n} \vee \overline{a_{n-1}}) \bigwedge_{1<i<n} ((\overline{x_i} \vee a_i) \wedge (\overline{a_{i-1}} \vee a_i) \wedge (\overline{x_i} \vee \overline{a_{i-1}}))$$

where $a_i (1 \leq i \leq n)$ are auxiliary variables. In general, such an encoding is called the sequential encoding of AMO. The basic idea of this encoding is to build sequentially a count-and-compare hardware circuit and then translate the circuit to CNF. It is easy to see that this encoding requires $3n-4$ clauses and $n-1$ auxiliary variables. In terms of number of auxiliary variables, this encoding is not optimal.

An encoding that uses fewer auxiliary variables is the logarithmic bitwise AMO encoding due to Frisch et al. [5, 6]. This encoding consists of the following clauses.

$$\overline{x_i} \vee a_k \ [\text{ or } \overline{a_k}\ ]$$

if bit k of the binary representation of $i-1$ is 1 [or 0], where $i = 1, 2, \ldots, n, k = 1, 2, \ldots, \lceil \log n \rceil$. Clearly, this encoding requires $\log n$ auxiliary variables and $n \log n$ clauses. Prestwich [2] used the logarithmic bitwise AMO encoding to remodel two problems: parity learning and Towers of Hanoi as STRIPS planning, and solved successfully them using a standard SAT local search algorithm. Although the logarithmic bitwise AMO encoding requires fewer auxiliary variables, it does require more clauses than the sequential encoding.

The paper presents a new SAT encoding of the AMO constraint that improves on the existing result. This encoding is implemented in a Cartesian product form. We consider n as an integer that is close to the product of two factors p and q, and combine AMO encodings of p and q variables into an AMO encoding of n variables. If encoding each sub-condition in a recursive way, such a 2-product encoding requires $2\sqrt{n}+O(\sqrt[4]{n})$ auxiliary variables and $2n+4\sqrt{n}+O(\sqrt[4]{n})$ clauses, which is fewer than the known best one. If we break down n into k parts, a 2-product encoding can be generalized into a k-product encoding. Our k-product encoding is not only of lower computation complexity, but also scalability. In the rough sense, the logarithmic bitwise AMO encoding can be considered as a special case of our k-product encoding. In addition, our k-product encoding preserves still arc-consistency. That is, as soon as one variable among n variables becomes true, unit propagation sets all other variables to false.

2 A new AMO encoding

Below we will present a new encoding of the Boolean at most one (AMO) constraint. Its basic principle is based on Cartesian products. That is, it uses a grid with n points, one for each variable, and constrains at most one row and at most one column to be selected, and the point at their intersection is the only variable allowed to be true. In details, we break down n into two parts: p and q. The product of p and q is equal to or greater and closest to n. In general, let $p = \lceil \sqrt{n} \rceil, q = \lceil n/p \rceil$, where $\lceil x \rceil$ denotes the rounding to the nearest integer that is greater than or equal to x. We make one point x_k in one dimension coordinate correspond to one point $\langle u_i, v_j \rangle$ in two dimension coordinate, where $x_k \in \{x_1, x_2, \ldots, x_n\}, u_i \in \{u_1, u_2, \ldots, u_p\}$ and $v_i \in \{v_1, v_2, \ldots, v_q\}$. In general, we have $pq > n$. In this case, in order to ensure that our mapping is one-to-one, we remove the extraneous grid points and select only n points from pq points. Assuming $n = (p-1)q + r$ and $1 \leq r \leq q$, we select the n points $\langle u_i, v_j \rangle$ with $(i-1)q+j \leq n, 1 \leq i < p$ and $1 \leq j \leq q$. For example, if n is 5, then $p = 3, q = 2$, and x_1, x_2, x_3, x_4, x_5 are mapped into $\langle u_1, v_1 \rangle, \langle u_1, v_2 \rangle, \langle u_2, v_1 \rangle, \langle u_2, v_2 \rangle, \langle u_3, v_1 \rangle$, respectively. In such a mapping, the proposition " at most one out of n points $x_1, x_2, \ldots, x_n$ is true " is equivalent to the proposition " at most one out of pq points $\langle u_i, v_j \rangle$ is true". Using the equivalent transform of this proposition, our AMO encoding may be defined recursively as

$$\mathrm{AMO}(X) \equiv \mathrm{AMO}(U) \wedge \mathrm{AMO}(V) \bigwedge_{1 \leq i \leq p, 1 \leq j \leq q}^{1 \leq k \leq n, k=(i-1)q+j} ((\overline{x_k} \vee u_i) \wedge (\overline{x_k} \vee v_j))$$

where $X = \{x_1, x_2, \ldots, x_n\}, U = \{u_1, u_2, \ldots, u_p\}, V = \{v_1, v_2, \ldots, v_q\}$, each element u_i in U and each element v_j in V are auxiliary variables. AMO(U) expresses that no more than one u_i in U is true. And AMO(V) expresses that no more than one v_j in V is true. Therefore, there is no more than one pair $\langle u_i, v_j \rangle$ whose u_i and v_j both are true. Thus, the constraint condition $(\overline{x_k} \vee u_i) \wedge (\overline{x_k} \vee v_j)$ implies that no more than x_k is true. Because this encoding is done in the product of 2 factors, we refer to such an encoding as the 2-product encoding of the AMO constraint. The encoding of AMO(U) and AMO(V) can be done by either a recursive or direct way. The direct way can be standard or sequential. The following is an example of the 2-product encoding by using the standard encoding to encode sub-constraints with $n = 5$.

Example 1. $n = 5$. We break down n into $p = 3, q = 2$.

$$\begin{aligned}
\mathrm{AMO}(U) &: (\overline{u_1} \vee \overline{u_2}) \wedge (\overline{u_1} \vee \overline{u_3}) \wedge (\overline{u_2} \vee \overline{u_3}) \\
\mathrm{AMO}(V) &: (\overline{v_1} \vee \overline{v_2}) \\
\mathrm{AMO}(X) &: (\overline{u_1} \vee \overline{u_2}) \wedge (\overline{u_1} \vee \overline{u_3}) \wedge (\overline{u_2} \vee \overline{u_3}) \wedge (\overline{v_1} \vee \overline{v_2}) \wedge \\
&\quad (\overline{x_1} \vee u_1) \wedge (\overline{x_1} \vee v_1) \wedge (\overline{x_2} \vee u_1) \wedge (\overline{x_2} \vee v_2) \wedge (\overline{x_3} \vee u_2) \wedge \\
&\quad (\overline{x_3} \vee v_1) \wedge (\overline{x_4} \vee u_2) \wedge (\overline{x_4} \vee v_2) \wedge (\overline{x_5} \vee u_3) \wedge (\overline{x_5} \vee v_1)
\end{aligned}$$

If using the sequential encoding mentioned above to encode two sub-constraints, based on the property of this encoding, it is not difficult to conclude that the resulting 2-product AMO encoding encoding is of the minimal clauses. In such a case, we have

Theorem 1. *If using the sequential encoding to encode sub-constraints* AMO(U) *and* AMO(V)*, where* $U = \{u_1, u_2, \ldots, u_p\}, V = \{v_1, v_2, \ldots, v_q\}$*,* $p = \lceil \sqrt{n} \rceil$*, and* $q = \lceil n/p \rceil$*, the 2-product encoding of AMO given above requires* $2n + (3p-4) + (3q-4) \approx 2n + 6\sqrt{n} - 8$ *clauses and at most* $4\sqrt{n}$ *auxiliary variables. The total number of literals appearing in the clauses is* $4n + 12\sqrt{n} - 16$.

This proof is trivial. When n is small, say $n \leq 6$, using the sequential encoding to encode sub-conditions is not a good choice. The resulting 2-product encoding requires more clauses and more auxiliary variables than the sequential encoding. Therefore, when $n \leq 20$, we suggest using the standard AMO encoding mentioned above to encode sub-conditions. In such a case, we have

Theorem 2. *If using the standard encoding to encode sub-constraints* $\mathrm{AMO}(U)$ *and* $\mathrm{AMO}(V)$*, where* $U = \{u_1, u_2, \ldots, u_p\}, V = \{v_1, v_2, \ldots, v_q\}$*,* $p = \lceil\sqrt{n}\rceil$*, and* $q = \lceil n/p \rceil$ *, the 2-product encoding of AMO given above requires* $2n+p(p-1)/2+q(q-1)/2 \approx 3n - \sqrt{n}$ *clauses and* $2\sqrt{n}$ *auxiliary variables. The total number of literals appearing in the clauses is* $6n - 2\sqrt{n}$*.*

This proof is trivial also. It is easy to see that the 2-product encoding based on the standard encoding requires fewer clauses and fewer auxiliary variables than the sequential encoding. In fact, the recursive way is also a good approach to reduce the number of clauses and auxiliary variables. The following is a theorem on a recursive way.

Theorem 3. *If encoding sub-constraints* $\mathrm{AMO}(U)$ *and* $\mathrm{AMO}(V)$ *in a recursive way, where* $U = \{u_1, u_2, \ldots, u_p\}, V = \{v_1, v_2, \ldots, v_q\}$*,* $p = \lceil\sqrt{n}\rceil$*, and* $q = \lceil n/p \rceil$*, the 2-product encoding of AMO given above requires* $2n + 4\sqrt{n} + O(\sqrt[4]{n})$ *clauses and* $2\sqrt{n} + O(\sqrt[4]{n})$ *auxiliary variables.*

Proof. Let $f(n)$ be the number of clauses required to encode n variables. By the property of this encoding, we have the following recurrence relation.

$$f(n) = 2n + f(p) + f(q).$$

Since $p \approx q \approx \sqrt{n}$, this recurrence relation can be simplified into

$$f(n) = 2n + 2f(\sqrt{n}).$$

By replacing repeatedly the recurrence term, we can obtain the following solution

$$f(n) = 2n + 4\sqrt{n} + 8\sqrt[4]{n} + 16\sqrt[8]{n} + \cdots\cdots = 2n + 4\sqrt{n} + O(\sqrt[4]{n}).$$

Let $g(n)$ be the number of auxiliary variables required to encode n variables. Then we have

$$g(n) = p + q + g(p) + g(q).$$

Replacing p and q with $\sqrt{n}$ yields

$$g(n) = 2\sqrt{n} + 2g(\sqrt{n}).$$

The solution to the recurrence term is approximately $2\sqrt{n} + O(\sqrt[4]{n})$. □

By this theorem, it is easy to see that the recursive 2-product encoding requires fewer clauses and fewer auxiliary variables than the previous 2-product encoding based on the sequential encoding. When n is small, say $n < 10$, the 2-product encoding based on the standard encoding is better. Notice, all the three 2-product encodings given above require more auxiliary variables than the bitwise AMO encoding. To reduce the number of auxiliary variables, we can extend the 2-product encoding to the k-product encoding. That is to say, we use a k-dimension grid with n points, one for each variable, and constrain at most one vertical plane at any coordinate axis to be selected, and the point at their intersection is the only variable allowed to be true. In details, we break down n into k parts: $p_1, p_2, \ldots, p_k$, where $p_1 = p_2 = \cdots = p_k = \sqrt[k]{n}$. We make a point in one dimension coordinate $X = \{x_1, x_2, \ldots, x_n\}$ correspond to a point in k

dimension coordinate $W_1 \times W_2 \times \ldots \times W_k$, where $W_i = \{w_1^i, w_2^i, \ldots, w_{p_i}^i\}, i = 1, 2, \ldots, k$. In order to ensure that the mapping is one-to-one, we remove the extraneous grid points and select only n points from $p_1 \times p_2 \times \ldots \times p_k$ points. Each element in W_i is considered as an auxiliary variable. Let $p_1 = p_2 = \cdots = p_k = p = \sqrt[k]{n}$, the k-product encoding of AMO may be defined as

$$\mathrm{AMO}(X) \equiv \bigwedge_{i=1}^{k} \mathrm{AMO}(W_i) \wedge \mathrm{map}(X, W_1, W_2, \ldots, W_k),$$

where $\mathrm{map}(X, W_1, W_2, \ldots, W_k)$ consists of the following clauses.

$$(\overline{x_i} \vee w_{a_1}^1) \wedge (\overline{x_i} \vee w_{a_2}^2) \wedge \ldots \wedge (\overline{x_i} \vee w_{a_k}^k)$$

i and $a_j (j = 1, 2, \ldots, k)$ satisfy the following relation.

$$i = (a_1 - 1)p^{k-1} + (a_2 - 1)p^{k-2} + \cdots + (a_{k-1} - 1)p + a_k$$

$$1 \leq i \leq n, 1 \leq a_j \leq p$$

This has nk clauses in total. Notice, the 2-product encoding is a special case of such an encoding. In the case of $k = 2$, if using the standard encoding to encode sub-constraints, we have Theorem 2. In the case of $k > 2$, similarly we have

Theorem 4. *If using the standard encoding to encode sub-constraints* $\mathrm{AMO}(W_i)$, $i = 1, 2, \ldots, k$, $k > 2$ *and each* p_i *is equal to* $p = \sqrt[k]{n}$ *the* k*-product encoding of AMO given above requires* $nk + k\sqrt[k]{n}(\sqrt[k]{n} - 1)/2$ *clauses and* $k\sqrt[k]{n}$ *auxiliary variables.*

Proof. Let $f(n)$ be the number of clauses required by the k-product encoding to encode n variables, $h(n)$ the number of clauses required by the standard encoding. Then we have

$$f(n) = nk + h(p_1) + h(p_2) + \cdots + h(p_k).$$

substituting $p = \sqrt[k]{n}$ for each p_i yields

$$f(n) = nk + kh(p).$$

Encoding p variables by the standard encoding requires $p(p - 1)/2$, Therefore we have

$$f(n) = nk + kp(p - 1)/2 = nk + k\sqrt[k]{n}(\sqrt[k]{n} - 1)/2.$$

The number of auxiliary variables required to encode n variables is

$$p_1 + p_2 + \cdots + p_k = kp = k\sqrt[k]{n}. \qquad \square$$

It is not difficult to see that the logarithmic bitwise AMO encoding [5] is a special case of the k-product encoding, i.e., the case of $k = \log n$. From Theorem 4, when $k = \log n$, i.e. $p = 2$, the number of clauses required by the k-product encoding is $n \log n + \log n$, which is $\log n$ more than the logarithmic bitwise AMO encoding [5]. This is because our W_i consists of two variables w_1^i and w_2^i , while the W_i in [5] consists of only one variable. If we reduce W_i to one variable, the k-product encoding is totally consist with the logarithmic bitwise AMO encoding. The k-product encoding has a significant advantage: it can trade off the computation complexities of clauses and auxiliary variables. Depending on various k, we can obtain encodings with different complexities. In terms of running time, for different SAT solvers, what is the optimal value of k? This should rely on real applications.

3 Empirical evaluation

Table 1. The number of clauses and auxiliary variables required by using various AMO encodings to encode AMO constraints of edge-matching problems.

Instance	n	k	2-product		sequential		bitwise		standard	
			#cls	#var	#cls	#var	#cls	#var	#cls	#var
em_14_7_3_cmp	144,48,7	401	134548	8358	136993	45798	359781	2931	3073893	0
em_12_2_4_cmp	100,40	280	67280	5040	68480	22920	159200	1880	1052400	0
em_11_3_4_cmp	81,36	234	45252	3780	46206	15480	107406	1566	570240	0
em_9_3_5_cmp	49,28	154	18256	1988	18494	6216	36652	868	136416	0
em_8_4_5_cmp	36,24	120	10608	1344	10752	3624	21312	672	58608	0

Table 2. Runtime (in seconds) required by CircleSAT to solve edge-matching problems based on various AMO encodings.

Instance	2-product	sequential	bitwise	standard
em_14_7_3_cmp	100.91	482.78	177.25	202.47
em_12_2_4_cmp	252.19	143.61	158.23	486.69
em_11_3_4_cmp	114.06	302.66	746.94	321.02
em_9_3_5_cmp	168.97	186.58	52.23	106.44
em_8_4_5_cmp	12.53	57.60	51.31	54.61
em_14_7_3_fbc	5.53	6.52	7.97	35.33
em_12_2_4_fbc	77.20	34.48	92.55	20.06
em_11_3_4_fbc	18.08	68.70	8.28	12.84
em_9_3_5_fbc	23.69	13.39	22.88	75.84
em_8_4_5_fbc	54.66	17.48	55.33	66.91

Recently, Frisch and Giannaros [10] carried out evaluation experiments with various encoding approaches, including 2-product AMO encoding, sequential AMO encoding, bitwise AMO encoding, standard AMO encoding and commander AMO encoding [11] etc. The instances tested are from the pigeonhole problem. In their experiments evaluating the performance of AMO encodings, the 2-product AMO encoding was the quickest [10].

Here we conducted another evaluation experiment. Unlike the instances used by Frisch and Giannaros, the instances used by us are from edge-matching problems submitted by Heule to SAT 2009 competition [7]. This is a popular puzzles, that traced back to 1890's. Edge-matching problems are formulated as follows. Given a set of pieces and a grid, the problem is whether we can place the pieces on the grid such that the edges of the connected pieces match. It is proved to be NP-complete.

Except for *em_8_4_5_cmp* and *em_8_4_5_fbc*, all the instances selected cannot be solved by any solver in SAT 2009 competition [12]. To be able to solve easily the instances, we decided to remodel them. The remodeling approach adopted by us is to translate the implicit one-on-one mapping into the explicit one-on-one mapping. Notice, the *cmp* and *fbc* family are encoded with the compact model, i.e., the implicit model. Using the explicit model adds the redundant clauses and requires more resource, but solve more easily the instance generated by it. We remodeled edge-matching problems using the following four different AMO encoding approaches: 2-product encoding, sequential encoding [1], logarithmic bitwise encoding [5] and standard encoding. The 2-product encoding here is Approach 2 given above (see Theorem 2), i.e., applying the standard encoding to encode sub-constraints. Because n here is small, the performance of our Approaches 1,2 and 3 is almost the same. Our Approach 4 is the same as the logarithmic bitwise encoding. Table 1 shows the performance of four different AMO encodings in terms of number of auxiliary variables and number of clauses. Columns n and k denote the number of variables in a constraint condition and the number of AMO constraints, respectively. The first data (144, 48, 7) in Column n represent that the instance has 3 types of AMO constraints with $n = 144, n = 48$ and $n = 7$. Columns #cls and #var denote the number of clauses and the number of auxiliary variables required by an AMO encoding, respectively. Table 1 omitted the result on the *fbc* family, since the *cmp* and *fbc* family have the same AMO constraints. As shown in Table 1, the 2-product encoding requires the minimal clauses, while the standard encoding does the maximal clauses. When the resource of memory is limited, the application of the standard encoding is limited easily.

We carried out the performance evaluation in terms of runtime under such a platform: Intel Core 2 Quad Q6600 CPU with speed of 2.40GHz and 2GB memory. We used the same SAT solver to solve all the instances obtained by various AMO encodings. With respect to the type of solvers, ours is the same as that used in [10], but differs from that used in [2] and [5]. The SAT solvers used in [2] and [5] both are based on on a local search. What we used is a revised version of CircleSAT [9], which is a conflict-driven DPLL complete solver based on PrecoSAT [8]. Table 2 shows the runtime required to solve each instance based on various AMO encodings with CircleSAT (the maximal number of learned clauses is limited to 150000). On average, the 2-product encoding was faster than other encodings like sequential, bitwise, and standard. In details, the 2-product encoding has six instances out of ten instances faster than sequential and bitwise, and seven instances faster than standard. From this experimental result, we can conclude that the solving speed depends not only on the effectiveness of the SAT encoding, but also the heuristic strategy used by a SAT solver. When the SAT encoding matches the heuristic strategy, the solving speed is certainly fast, especially for satisfiable problems.

4 Conclusions

We have developed four versions for the AMO encoding, each of which is effective. The version with the minimal clauses is the recursive 2-product encoding. This encoding requires $2n + 4\sqrt{n} + O(\sqrt[4]{n})$ clauses. Furthermore, using our encodings, unit propagations achieves arc-consistency. Sinz [1] has pointed out that each clausal encoding of the AMO condition requires at least n clauses. In this sense, this 2-product encoding is also optimal. Furthermore, we conjecture that this result has reached the best lower bound on the number of required clauses for any AMO encoding. An interesting topic might be to prove that this is the tightest lower bound. This will remain as an open problem.

References

1. Sinz, C.: Towards an optimal CNF encoding of Boolean cardinality constraints, Proceedings of Principles and Practice of Constraint Programming (CP), 827–831 (2005)
2. Prestwich, S.: Variable dependency in local search: prevention is better than cure. Proceedings of Theory and Applications of Satisfiability Testing (SAT), 107–120 (2007)
3. Argelich, J., Cabiscol, A., Lynce, I., Manya, F.: Sequential encodings from Max-CSP into partial Max-SAT, Proceedings of Theory and Applications of Satisfiability Testing (SAT), 161–166 (2009)
4. Porschen, S., Schmidt, T., Speckenmeyer, E.: On some aspects of mixed Horn formulas, Proceedings of Theory and Applications of Satisfiability Testing (SAT), 86–100 (2009)
5. Frisch, A. M., Peugniez, T. J., Doggett, A. J., Nightingale, P. W.: Solving non-Boolean satisfiability problems with stochastic local search: a comparison of encodings, Journal of Automated Reasoning, 143–179 (2005)
6. Frisch, A. M., Peugniez, T. J.: Solving non-Boolean satisfiability problems with stochastic local search, Proceedings of the 17th International Joint Conf. on Artificial Intelligence, 282–288 (2001)
7. Heule, M.: Solving edge-matching problems with satisfiability solvers, SAT 2009 competitive events booklet, 69–82 (2009).
8. Armin, B.: P{re,i}coSAT@SC'09, SAT 2009 competitive events booklet, p.41 (2009)
9. Chen,J.C.: CircleSAT for SAT-Race 2010, http://baldur.iti.uka.de/sat-race-2010/descriptions/solver_7.pdf.
10. Frisch, A., Giannaros, P.A.: SAT encoding of Boolean cardinality, some old, some new, some fast, some slow, Manuscript (2010)
11. Klieber, W., Kwon, G.: Efficient CNF encoding for selecting 1 from N objects, the Fourth Workshop on Constraints in Formal Verification (CFV), (2007)
12. SAT'09 Competition Homepage,http://www.cril.univ-artois.fr/SAT09/